CptS 427.01 – Cyber Security of Wireless and Distributed Systems (Spring 2024)

Due 4/7/2024 @ 11:59 PM PST via Canvas

[ ] Submit a single ZIP file containing your answer file and any source code you wrote to derive your answers.

Assignment #4 - Passwords

Every modern operating system has a method of authenticating users. This series of practical exercises will focus on analysis of different password files based on an imaginary specification.

Linux distributions use a similar authentication scheme (e.g., /etc/passwd and /etc/shadow files). The Windows® operating systems use a local SAM file for authenticating local users or Active Directory for centrally managed authentication. Regardless of the design and implementation, the key requirement of any password-based authentication scheme remains the same:

"Does the operating system recognize the given username and password?"

1. Password File Requirements

Early implementations of file-based password authenticators were vulnerable to disclosing secrets to hackers.

The itemized list below are the requirements of my imaginary authentication scheme. You will need these details to understand the format of all the password files in these practical exercises.

1. The password file must be self-contained.
2. The password file must use ASCII encoding.
3. The password file will contain one line per user.
4. Each line must provide storage for the following data (in order):
    1. User ID
    2. Group ID
    3. Username
    4. Email address
    5. Password
    6. Last authentication date
    7. Number of failed logins

2. Word List

MIT has published a list of 10,000 words (https://www.mit.edu/~ecprice/wordlist.10000), which I used to generate all the password entries for this assignment.

Please use the local copy of `wordList.txt` from Canvas to ensure we are both working from the same list of words.

3. Salted Hashes

Obviously, plaintext password storage is a bad idea. One of the earliest methods of protecting the confidentiality of passwords was to encrypt them. However, using the same key to encrypt the same plaintext password resulted in the same ciphertext. Weak passwords could be guessed, then the key used to encrypt the weak password is compromised.

The compromised key can then be used to decrypt stronger passwords.

To solve this problem, additional entropy is needed. A *salt* is added to the plaintext password so the resulting ciphertext is different. For example, two users have both chosen the weak password of "hello". Both instances of "hello" are salted with different words, resulting in different ciphertext. (In this case, we use the md5 algorithm for simplicity—but the result is the same.)

plaintext	ciphertext
hello	5d41402abc4b2a76b9719d911017c592
hello+there	c6f7c372641dd25e0fddf0215375561f
hello+world	fc5e038d38a57032085441e7fe7010b0

(Use "md5 -s [plaintext]" to recreate these results but omit the "+" in the plaintext. It is just there to illustrate the password+salt relationship.)

But if we use the same salt for every password—we have not mitigated the vulnerability; we've just made the implementation more complicated. (Don't laugh—a vendor you've probably heard of made this mistake.) Complexity is the enemy of security.

4. Password.txt

The password field of `Password.txt` is simply the output of the function md5(password+salt). The plus sign indicates the concatenation operation and should not be included in the input.

For example, if the password is "Star" and the salt is "Wars", you would use md5(StarWars) not md5(Star+Wars).

Analyze the Password.txt file on Canvas to answer the following questions:

1. What is the salt that was added to each password?
2. Which user had the most failed logins, and how many failed logins did they have?
3. Which user has not logged in for the longest time?

Each password and salt value comes from `wordlist.txt`.

5. Unit Tests

Password	Salt	MD5 Hash
		d41d8cd98f00b204e9800998ecf8427e
abandoned		81f3d2a447d9c5579b446ff048827de1
idaho	saturn	81d4ce3fd1613924ed42bb4928c7e645

In the first example no password and no salt are specified.

In the second example a password is specified, but no salt is specified.

In the third example, both a password and salt are specified.

Any program you write should generate the same MD5 sums as these unit tests.